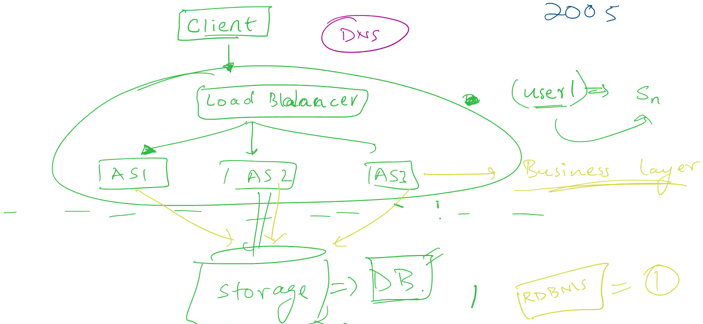
Sharding

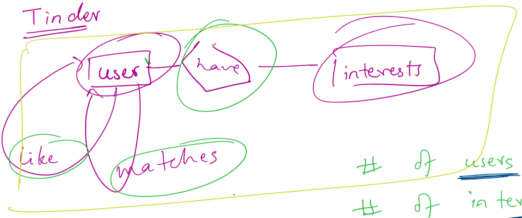
Tuesday, 1 December 2020 9:00 PM

- SQL
- Shard Unstructured data:
  Google search Typeahead
- Problem statement: Big Table (1st No SQL System 2005)


Client
DNS
Load Balancer
AS1
AS2
AS3
Business layer
(user1) ⇒ Sn
storage
⇒ DB
RDBMS = ①

- read heavy (cache)
- write heavy (shard)

Tinder


user
have
interests
like
matches

# of users ↑

# of interested

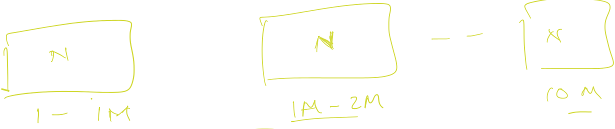
→ RAM

→ HDD

→ clock speed (3.2 GHz) ⇒ 5-6 GHz

→ # of cores

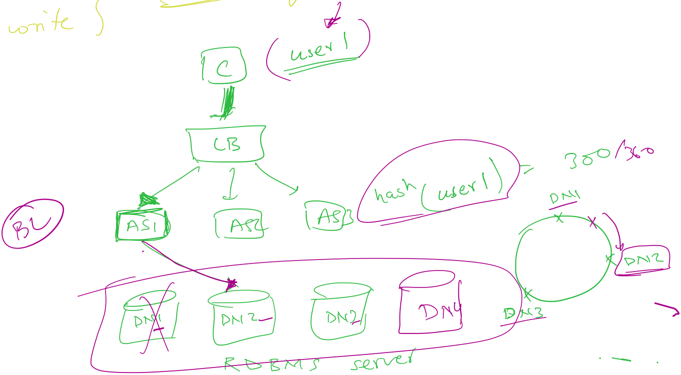
user 1 — — — user 10M


N
N
X
1 – 1M
1M – 2M
10 M

user-id = key.

- schema is same ✓
- but data is unique. ✓
- read
- write

Hot key


user1
C
LB
BL
AS1
AS2
AS3
hash(user1) = 300/360
DN1
DN2
DN3
DN1
DN2
DN2
DN4
RDBMS server

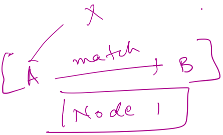
⊙ routing the request is on App server

⊙ manul data movement is required

- Additn of node
- remaval of a node

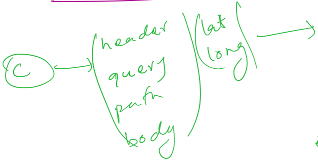
⊙ App Node needs of aware of

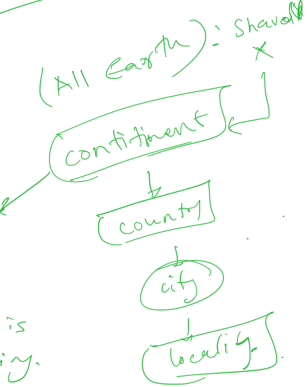
health of data nodes.


X
match
A
B
Node 1

(hash(user_id)

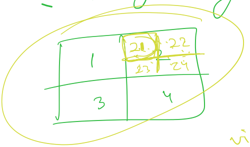
(hash(location)


C
header
query
path
body
lat
long

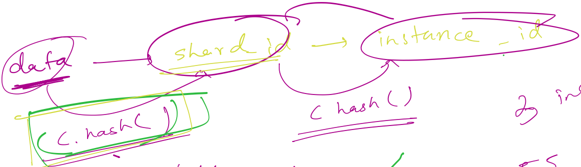

(All Earth): Shard
X
continent
country
city
locality

auto-sharding

amount of data is defining my sharding.


1
21
22
23
24
3
4

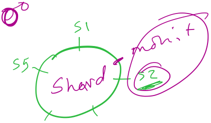
each shard will be physically persisted on a virtually / physical instance

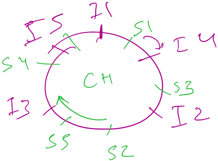

data
shard_id
instance_id
(C.hash()
(hash()

2 instance => 5 in

hash(mohit) = 50/360

= 50°


S1
S5
Shard
mohit
S2


I1
S5
S1
I4
S4
CH
S3
I3
I2
S5
S2

S4 S3

CH

~~Ring~~

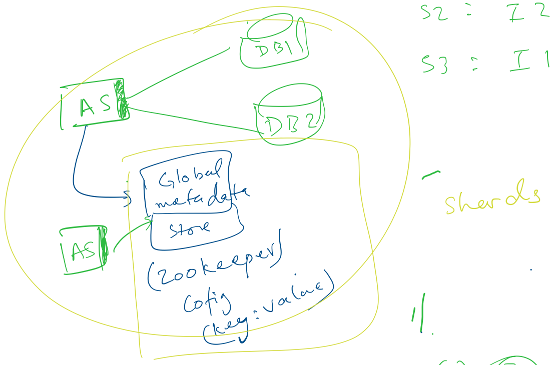
Hash Map

S1 : I1

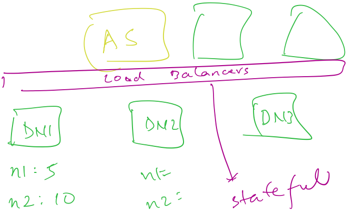
S2 : I2

S3 : I1

1:1



AS
DB1
DB2
Global metadata store
AS
(zookeeper)
cofig
(key:value)
shards



AS
Load Balancers
DN1
DN2
DN3
n1 : 5
n2 : 10
n1=
n2=
stateful

add(); ③

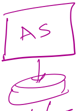
① set N1(n1);

② set N2(n2);

master

```
set Num1
set Num2
add() {
return num1+num2;
}
```

single m/c



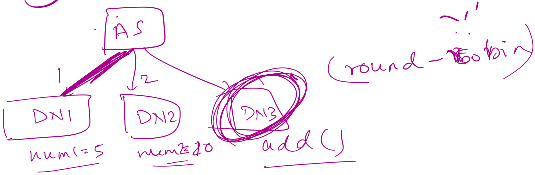
C: ① app.setNum1(5

② app.setNum2(10);

③ ~~print~~(app.add());

15

15



AS
1
2
DN1
DN2
DN3
num1=5
num2=10
add()
(round-robin)

Google Search Type Ahead

① strict prefix match. Mich Mick

② order by popularity. (Top 5)

③ personalization not supported.

- * Top 5 results
- * update the freq.

⊙ Single Machine : Trie

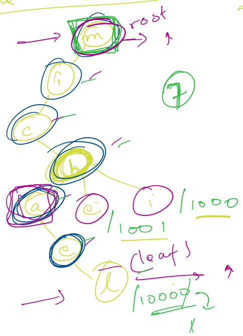
match

q -

m

mich

① → ①



root
m
i
c
h
a
e
i
e
l
7
/1000
/1001
(leaf)
/10002

mich'
michigan
michele

reads = write

top5

O(n)

latency : O(n) X

1M ⇒ 1M+1

top5 O(1)

heap

Node {
c c i/1000'
freq -
is last -
Node[] heap
}

⊙ writing

write throughput

① Caching (Reads) =

② Batch update ( )

③ Lazy writing (1 min)

④ for (1 hr) keep update on the leaf node

(Roger Fedder) ⇒

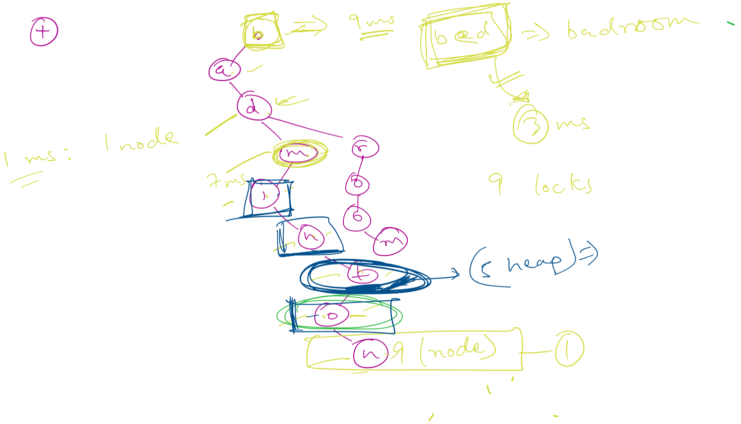
④ size 2k HashMap( )

phrase :

Roger : 1 2 3 — 1000

writes from lr reads

④



b ⇒ 9ms
bed ⇒ bedroom
3 ms
9 locks
1 ms : 1 node
7ms
a
d
m
i
n
t
o
n 9 (node)
1
r
o
o
m
(5 heap) ⇒

hash ⇒ size

④ (Sampling) (Donald Trump) =

(Donald Duck)

(99) → (150)

Time Decay : (Mar ⇒ Maradona :

Maruti :)

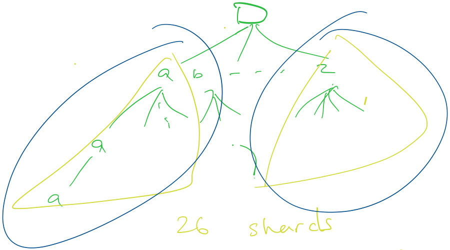
10000 → 9000

$f = \frac{t}{1+d} =$

$= \quad d = 0.5 - 0.7$

cron job: every day / hour / week

How do we persist?



① amount of data was huge,

26 shards

(a – z) => 26 char

=> A, S (Hot shard)

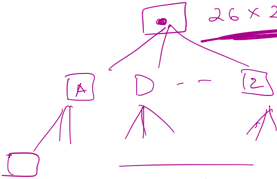
=> Z, X, W

Replicatⁿ => reads.

shard : 1 char
2 char
3 chars

A = 1 TB → 1 instⁿ

as
as
as
as
S1
S2
S3



26 × 26 × 26 TB –

26 × 26 TB –

26 TB –

1 TB –

Insta

3

26

3

4

① single instance cannot hold the data that's why we ~~have~~ more instances.

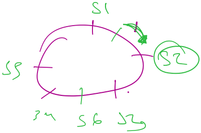
① (Replicas) are there ∵ we have hot shard i.e throughput of 1 type of request is huge.

Donald : ⟶ 1

Justin : ⟶ 1 + 1

③

(aaa), aab – – zzz


S1
S2
S5
S4
S6
S3

CH ( )

hash(ass) →

user

assissination

② as

hash Map

128 × 128 × 128

aaa : S1

bbb : S2

Scaler

- User
- lectures
- Assignments
- Contest
- Practise :

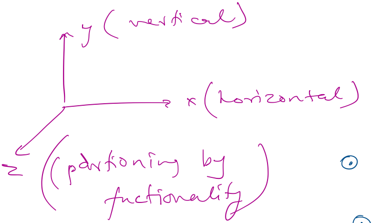
RDBMS

monolith

microservices

z - scaling

cube architecture :


y (vertical)
x (horizontal)
z ((partioning by fuctionality)

- Aadhar
- flipkart =( )

Joe Biden

Joe B

100

99

1